

# Learning Word Embeddings from Intrinsic and Extrinsic Views

Jifan Chen<sup>1</sup>, Kan Chen<sup>1</sup>, Xipeng Qiu<sup>1</sup>, Qi Zhang<sup>1</sup>, Xuanjing Huang<sup>1</sup>, Zheng Zhang<sup>2</sup>

<sup>1</sup> Shanghai Key Laboratory of Data Science

School of Computer Science, Fudan University

{jfchen14, kchen13, xpqiu, qz, xjhuang}@fudan.edu.cn

<sup>2</sup> Department of Computer Science, New York University Shanghai

zz@nyu.edu

## Abstract

While word embeddings are currently predominant for natural language processing, most of existing models learn them solely from their contexts. However, these context-based word embeddings are limited since not all words' meaning can be learned based on only context. Moreover, it is also difficult to learn the representation of the rare words due to data sparsity problem. In this work, we address these issues by learning the representations of words by integrating their intrinsic (descriptive) and extrinsic (contextual) information. To prove the effectiveness of our model, we evaluate it on four tasks, including *word similarity*, *reverse dictionaries*, *Wiki link prediction*, and *document classification*. Experiment results show that our model is powerful in both word and document modeling.

## 1 Introduction

Word embeddings, also known as distributed word representations, can better capture a large number of syntactic and semantic word relationships (Mikolov et al., 2013b), and have been proven to be useful in many NLP tasks, such as language modeling (Bengio et al., 2006), parsing (Socher et al., 2013), relation extraction (Riedel et al., 2013), discourse relation detection (Chen et al., 2016) and so on.

Most of the existing methods for learning word embeddings try to predict the current word using its context either through a neural network (Bengio et al., 2006; Mikolov et al., 2010) or a simple log-linear model (Mikolov et al., 2013a). However, such purely data-driven approaches suffer a number of problems in practice, for example:

1. With limited length of context window, some words with entirely different meanings may share very similar context. Embeddings of such words become indistinguishable (Hill et al., 2016). For example, “old” and “new”.
2. Context-based models are unable to generate reasonable embeddings for rare words due to data sparsity problem.
3. The meaning of some words, especially for some named entities, places, people, are hard to be learned only by their contexts.

To address the above issues, we refer to the cognitive process of human learning. For a language unit (word or phrase), we can learn its meaning from two kinds of information:

**Intrinsic information** The intrinsic information can be a concise explanation (called definition or descriptions) to the meaning of a language unit. Such explanations often offer deeper understandings to a language unit.

**Extrinsic information** The extrinsic information can be contexts which surround a language unit and help to determine its interpretation.

In this paper, we improve the word embedding by utilizing both intrinsic (descriptive) and extrinsic (contextual) information. More specifically, given a word (or a phrase) and its description (or definition),

we first generate its intrinsic representation from the description, and then learn the final representations with its context. Crucially, these two sources of information are fused within one unified framework. While many implementation choices exist, as a proof of concept, we simply extend the well-known Skip-gram (Mikolov et al., 2013a). In our model, Skip-gram is used in two different ways but within one uniformly defined loss function: the first is to compute a representation of a word’s definition (description), and the second is to compute in-context word representation. We use Wikipedia pages in our experiments, since words described by Wikipedia are often some named entities, and they are more likely to face data sparsity problem. In addition, Wikipedia contains abundant intrinsic information of words which is helpful for training.

The main contribution of this work is that, we integrate intrinsic and extrinsic information to learn the distributed representations of words. The experiments show that word embeddings learned by our model are significantly better than the previous models on four different tasks.

## 2 Proposed Model

We start by discussing the skip-gram model (Mikolov et al., 2013a) which is a well known method for learning word embedding, our proposed model is an extension of their work.

### 2.1 Skip-gram Model

Given a word sequence  $x_1, x_2, \dots, x_N$ , the task of skip-gram is to predict the surrounding words within a certain distance based on the current one. More formally, the objective of skip-gram model is to maximize the following average log probability:

$$L = \frac{1}{T} \sum_{t=1}^T \sum_{-c \leq j \leq c, j \neq 0} \log p(w_{t+j}|w_t), \quad (1)$$

where  $c$  is the size of training window, and the probability  $p(w_{t+j}|w_t)$  is defined as:

$$p(w_{t+j}|w_t) = \frac{\exp(v_{w_t}^T v'_{w_{t+j}})}{\sum_{w=1}^W \exp(v_{w_t}^T v'_w)}, \quad (2)$$

where  $v_w$  and  $v'_w$  are the “word” and “context” vector representations of  $w$ , and  $W$  is the number of words in the vocabulary.

Since the cost of computing  $\nabla \log p(w_{t+j}|w_t)$  is proportional to the vocabulary size  $W$ , this formation is often impractical. In practice, an efficient alternative way is to use Negative Sampling (Mikolov et al., 2013a), which leads to the following objective:

$$L = \frac{1}{T} \sum_{t=1}^T \sum_{-c \leq j \leq c, j \neq 0} \log \sigma(v_{w_t}^T v'_{w_{t+j}}) + \sum_{i=1}^k \mathbb{E}_{w_i \sim P_n(w)} [\log \sigma(-v_{w_t}^T v'_{w_i})], \quad (3)$$

where  $k$  is the number of negative samples,  $\sigma$  denotes the sigmoid function, and  $P_n(w)$  is a noise distribution which is often set to the unigram distribution  $U(w)$  raised to the 3/4rd power. The task of this objective is to distinguish the target word from the noise distribution  $P_n(w)$  using logistic regression. In the following of this paper, we will use (3) instead of (1) as the objective function.

### 2.2 Definition Enriched Word Embedding

We assume a collection of document  $C = (D_1, D_2, \dots, D_C)$ , which offers definitions or descriptions to the corresponding word, and each document  $D_i$  consists of a sequence of words  $D_i = (w_1, w_2, \dots, w_N)$ , and  $R(w_i) = D_j$  if  $w_i$  has a link (or reference) to another document  $D_j$ .

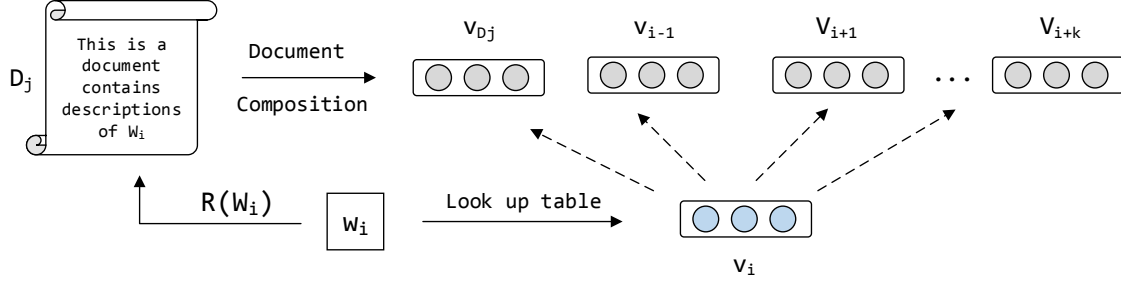


Figure 1: The framework of our model.

### 2.2.1 Intrinsic Representation from Definition

There exists many commonly used compositional methods to construct document embedding using word embeddings, including simple word embedding addition, multiplication (Mitchell and Lapata, 2010), Recurrent Neural Network (Hochreiter and Schmidhuber, 1997), Convolution Neural Network (Hu et al., 2014), Recursive Neural Network (Socher et al., 2011) and so on. It is reported in the work of Hill et al. (2015) and Blacoe and Lapata (2012), that although the word embedding addition is simple, it still achieves comparable performance on several tasks compared with neural networks like RNN which is much more time consuming since it involves lots of non-linear computations. Concerning with the training efficiency, we in this work use the simple word embedding addition to represent document embedding:

$$v_{d_i} = \sum_{w_j \in D_i} \alpha_j \cdot v_{w_j}, \quad (4)$$

where  $v_{w_j}$  is the embedding of word  $w_j$ , and  $\alpha_j$  is its corresponding weight. In the simplest case, all the weights are equal to 1, while the weights can be also set to TF-IDF weights to avoid the affects of the stop words.

### 2.2.2 Learning Word Embedding with its Definition

We implement our model by adding a loss term to Skip-gram, and the learning framework is shown in Figure 1. For a word  $w_t$  that has a link to its definition, the objective function is:

$$L = \sum_{-c \leq j \leq c, j \neq 0} \log \sigma(v_{w_t}^T v'_{w_{t+j}}) + \sum_{i=1}^k \mathbb{E}_{w_i \sim P_n(w)} [\log \sigma(-v_{w_t}^T v'_{w_i})] + \log \sigma(v_{w_t}^T v_{R(w_t)}), \quad (5)$$

and the words without external descriptions are computed as Skip-gram (i.e. without the last term in the equation). Notice that for a word with a link or reference, the change of its embedding also modifies the embedding of its description document, and therefore indirectly changes other word embeddings in that document. We call our generated embeddings as **Definition Enriched Word Embedding (DEWE)**.

## 3 Experiment

In this section, we first describe how our training set is constructed, and report experimental results that test the effectiveness and robustness of our model.

### 3.1 Dataset

We construct our training set by extracting wiki pages from Wikipedia using the API provided<sup>1</sup>. The Wikipedia is organized in tree structure by document categories<sup>2</sup>, and each child node in the tree is one of the subcategories of its parent node. Since the pages in the same category are likely to reference each other, we start by extracting the pages in “Computer Science” category, and expand the dataset by adding the pages from its subcategories. We limit the search depth in the tree to 3, because the deeper the search depth is, the less relevant the pages are. After combing the same pages, our dataset contains 21,234 unique pages and approximately 150,000 unique words.

<sup>1</sup>[https://www.mediawiki.org/wiki/API:Main\\_page](https://www.mediawiki.org/wiki/API:Main_page)

<sup>2</sup><https://en.wikipedia.org/wiki/Portal:Contents/Categories>

Table 1: Top 5 nearest neighbors of the given word.

Word	Microsoft
Skip-gram DEWE	1. GroupWise 2. Apple Inc. 3. Novell 4. AdWords 5. Sun 1. IBM 2. Symantec 3. MS-DOS 4. Google 5. Internet Explorer
Word	Bill Gates
Skip-gram DEWE	1. Steve Jobs 2. Zuckerber 3. Reid 4. Poole 5. Bryant 1. Steve Jobs 2. Microsoft 3. Zuckerber 4. IBM 5. Windows
Word	Database
Skip-gram DEWE	1. NVD 2. Object database 3. Mobile device 4. Computer network 5. Document 1. DBMS 2. SQL 3. Data 4. Software 5. Microsoft Excel
Word	WordNet
Skip-gram DEWE	1. DSSSL 2. Metamodeling 3. Leet 4. Thesaurus 5. RMIAS 1. Synsets 2. Tatoeba 3. BabelNet 4. lexical 5. dictionaries

### 3.1.1 Data Preprocessing

To make the training for the words with links sufficient enough, we enforce the following invariant: for all occurrences of a word, its link status is consistent. That is, a word either has no link at all, or it points to the same document. In practice, that means we add a link to new occurrences for a word if it has a link before. By enforcing this invariant, approximately 1% words in the training set have a link to the definition pages.

### 3.2 Implementation Details

For the implementation of our model, we did not conduct a hyper-parameter search on any validation dataset, instead, we choose the standard parameters that performs well based on the previous research. We choose the size of context window to be 3, the number of negative samples in the negative sampling part to be 5. We use the first 100 words in a document to construct its document embedding, as in Wikipedia, a page often contains thousands of words and the beginning part usually offers definition to the word it describes. We ignore the words which appear less than 10 times in the training set, and we use the TF-IDF weight to compose the document embedding. We set the embedding dimension to be 50, the batch size to be 10, the initial learning rate to be 0.5, and the total training epoch is set to 10. The model is implemented with Theano (Bergstra et al., 2010) and trained with mini-batch on GPU, and the learning rate is decreased by the proportion of the trained words and the total words (Řehůřek and Sojka, 2010). The parameters are kept the same for all of the experiments.

### 3.3 Competitors

For the task of word similarity, we compare our model with Skip-gram and Glove (Pennington et al., 2014) which is also one of the state-of-the-art word embedding learning method, to see how the definitions affect the generated word embeddings. For the task of reverse dictionary, there exists some commercial systems of reverse dictionary like **Onelook.com**<sup>3</sup>, and recently Hill et al. (2015) also implemented a reverse dictionary using word embeddings. However, the dataset they used to build their model are ordinary dictionaries, while our model is trained with Wikipedia which contains many entities, thus making it hard to do the comparison. Based on this, we compare our model with Skip-gram to see how much improvement we can achieve by adding the loss term. More specifically, we compose the document embedding using word embeddings trained with Skip-gram by point wise addition and multiplication, namely **Skip-gram Add** and **Skip-gram Mult**, which are established ways to construct document embedding. For link prediction and document classification, besides Skip-gram, we also compare with Paragraph Vector (**PV**) (Le and Mikolov, 2014) which is a unsupervised method to represent sentence and paragraph.

<sup>3</sup><http://www.onelook.com/reverse-dictionary.shtml>

Table 2: Performance of our proposed methods on several word similarity datasets.

	WS-353	MEN	MTurk-771	YP-130	SimLex-999
Skip-gram	44.57%	37.08%	31.95%	4.25%	17.88%
Glove	<b>45.35%</b>	32.93%	35.29%	8.64%	20.04%
DEWE	43.97%	<b>38.47%</b>	<b>36.93%</b>	<b>13.82%</b>	<b>21.46%</b>

### 3.4 Word Similarity

Although it is hard to evaluate word embeddings directly (Faruqui et al., 2016), we in this section evaluate the generated word embeddings of our model to see what characteristics are captured. The evaluation is divided into two parts, the first part is a qualitative analysis about the words with references, and the second part is a word similarity evaluation about the words without references.

#### 3.4.1 Qualitative Analysis

Table 1 shows the nearest neighbors of some entity names we selected from the training set. See “Microsoft” and “Bill Gates” first, it is easy to observe that the nearest neighbors returns by Skip-gram only contains company names and person names respectively. This is because Skip-gram is a context-based method, which captures word co-occurrence statistics. It means words are similar only if they share similar context. However, seen from the results of our model, **DEWE** returns “MS-DOS” and “Internet Explorer” for “Microsoft”, and returns “Microsoft” and “Windows” for “Bill Gates”, which are actually relevant while they are unlikely to appears in similar context. Such results demonstrate the definition “enrichment” to the context-based word embeddings.

Next, we consider “Database” and “WordNet”. Since they are rare words in the training set, and there are few words in the training set sharing similar context with this two words, so the Skip-gram model fails to return relevant words. In such situation, our model still returns satisfied results like “DBMS” and “SQL” to “Database”, “Synsets” and “lexical” to “WordNet”, which come from the definition. It shows when facing data sparsity problem, the definition “enrichment” is more significant.

#### 3.4.2 Word Similarity of Ordinary Words

Word similarity between word pairs can be generally divided into two aspects, *concept similarity* and *word association* (Hill et al., 2016). The difference between the two aspects can be exemplified by word pairs like {coast, shore} and {clothes, closet}. Coast and shore are similar since their definitions are similar, while clothes is associated with closet since they frequently occur together in space and language. We in this section conduct experiment on several commonly used word similarity datasets including WS-353 (Finkelstein et al., 2001), MEN (Bruni et al., 2012), MTurk-771 (Halawi et al., 2012), YP-130 (Yang and Powers, 2006), SimLex-999 (Hill et al., 2016), which contain only ordinary words, to see which aspect is captured by our model. The experiment is done by using the web tool provided by Faruqui and Dyer (2014).

The results are shown in Table 2. As we can see from the table, **DEWE** significantly outperforms Skip-gram on YP-130 and SimLex-999. This result is impressive, since SimLex-999 is a dataset especially designed for *concept similarity*, and YP-130 defines similarity based on WordNet taxonomy, it is obvious that although such ordinary words do not have a link to their definitions, their meanings are still enriched by the definitions of other words!

WS-353, MEN, and MTurk-771 consider both *association* and *concept similarity* as word similarity, while human annotators show different predilections to this two kind of similarities in different datasets. Our model outperforms Skip-gram on MEN and MTurk-771, while achieves a lower performance on WS-353. It demonstrates that our word embedding can hold both *word association* and *concept similarity* at the same time.

Glove is also a context based method, which leads to its less satisfied performance on YP-130 and SimLex-999. However, since it combines the global word co-occurrence statistics as well as local context window information, it generally performs better than Skip-gram.

Table 3: The difference between the Wikipedia definition and the manually-written definition.

Word	Test set	Description
Blog	Wikipedia Definition	A blog is a discussion or informational site published on the World Wide Web consisting of discrete entries ("posts") typically displayed in reverse chronological order...
	Human Description	Websites where owners can upload essays and pictures and interact with other users
Microsoft	Wikipedia Definition	Microsoft Corporation is an American multinational technology company in Redmond, Washington, that develops, manufactures, licenses, supports and sells computer software, consumer electronics and personal computers
	Human Description	A technology company founded by Bill Gates, most famous for its Windows operating systems

### 3.5 Reverse Dictionary

As opposed to the regular dictionary that maps words to its definitions, reverse dictionary performs the converse mapping. It addresses the "word is on the tip of my tongue, but I can not quite remember it" problem by returning a set of candidate words given a definition or description of a word (Zock and Bilac, 2004; Shaw et al., 2013). We implement this by first computing the document embedding of the given description, and then return the words corresponding to the embeddings which are closest to that document embedding using cosine similarity.

To test the effectiveness and robustness of our model, we design two types of the test data. The first one contains 200 randomly selected documents in the training dataset, which have been **seen** during training. The second one contains 100 **unseen** descriptions write by humans which are quite different from those in the training set. To do so, we randomly select 100 page names among the top 1000 most frequent page names appear in the training set. We then ask 10 graduate students of Computer Science Department to write descriptions towards these page names based on their own understandings. To ensure the quality of these description, we also ask another two graduate students to predict 5 words according to the descriptions, if the target name is failed to be predicted by one of the checkers, the original participant is asked to rewrite the description.

An example of the two kind of descriptions towards the same word is shown in Table 3, as we can see from the table that the definition in Wikipedia is much more formal and longer than the manually-written descriptions, and the human writers tend to describe entities using its most distinguishable characteristics.

#### 3.5.1 Results

The performances of all the models are shown in Table 4. Here we follow the measurements used in Hill et al. (2015): the *median rank* of the correct answer (lower better), the proportion of the correct answer appearing in top 10/100 words (*accuracy@10/100* higher better), and the standard variance of the rank of the correct answer (*rank variance* lower better). We do not report the performance of **Skip mult**, since it almost fails to return the corresponding word given any descriptions.

Seen from the results of the original Wikipedia Definition first, the first highlight is that our **DEWE** model outperforms Skip-gram significantly. In the case of **DEWE+tfidf**, it returns 90% of the correct words among the top-ten candidates with median rank equals to 2. It demonstrates that despite the definition is a relatively long document containing severals sentences, our model still has the ability to map it to the corresponding word by using the simple add operation.

There is also an interesting observation that although our DEWE model is trained with TF-IDF weights, it still achieves fairly good performances (better than **Skip-gram+tfidf**) without using TF-IDF weights in the test phase. It shows that our model also has the ability of fault tolerance, since the stop words can be seen as the noisy data in the description.

Seen from the results of the Manually-written Descriptions, we can find that the performances of

Table 4: The performance of different models on reverse dictionary. Notice that our **DEWE** model is trained with TF-IDF weight, so **DEWE+tfidf** here denotes using TF-IDF weights in the test phase.

Test set	Wikipedia Definition			Manually-written Description		
Skip add	231	6.5%/27.5%	243	425	2.0%/10.0%	310
Skip add+tfidf	110	26.5%/68.5%	178	257	8.0%/25.0%	256
DEWE	75	36.5%/78.5%	141	186	16.0%/49.0%	227
DEWE+tfidf	<b>2</b>	<b>90.0%/99.0%</b>	<b>3</b>	<b>112</b>	<b>35.0%/ 67.0%</b>	<b>200</b>

|      *median rank*   *accuracy@10/100*   *rank variance*      |

all methods decrease a lot. This result is reasonable, as we can see in Table 3, the manually-written descriptions are much shorter than the training document which contains 100 words in our experiment. Moreover, the manually-written descriptions may describe the words from a quite different perspective based on some common senses, which are totally different from the definitions in Wikipedia. Facing such situation, the Skip-gram model almost fails to return correct answers, while our model still achieves a satisfied performance. This results shows that our model can also generalize well to new descriptions which are never seen before.

### 3.5.2 Qualitative Analysis

We list some example outputs of manually-written descriptions from different models in Table 5. As we can see from the table, given the description, the Skip-gram fails to return any relevant words which is consistent with the results in Table 4. While our model not only returns the correct answer, but also returns other relevant names according to the description.

From the results, we can also see how the TF-IDF weights affect the result. In the first case, both **DEWE** and **DEWE+tfidf** return “Microsoft”, however, **DEWE+tfidf** returns “Bill Gates” ahead of “Microsoft”, it demonstrate that “Bill Gates” has a relatively high weight in the document, so it has a large impact to the result. While in the second case, we can see the effectiveness of the TF-IDF weights. It is easy to observe that **DEWE** is heavily affected by “media”, so that the words it returns are all about media, and it fails to return “Facebook”. However, with word weights counted, our model successfully returns the correct answer and some other relevant companies.

### 3.6 Link Prediction and Document Classification

In Wikipedia, there is never a completely isolated document. A wiki entry contains multiple referencing hyperlinks to other related entries which provide further understandings to this entry. Given a Wikipedia document, the task of link prediction is to find other documents which have a link to this document.

Table 5: Top 5 candidate words returned for manually-written descriptions.

Description	A technology company founded by Bill Gates most famous for its Windows operating systems
Skip add	1. AlterGeo 2. Project Athena 3. Milkymist 4. Lookout 5. Fabasoft
Skip add+tfidf	1. Bill Gates 2. Lookout 3. Apple Newton 4. Taligent 5. Ultima
DEWE	1. Microsoft 2. Wintel 3. Linux 4. IBM 5. Nimbula
DEWE+tfidf	1. Bill Gates 2. Microsoft 3. Apple 4. IBM 5. Windows NT
Description	The largest social media company in the world founded by Mark Zuckerberg
Skip add	1. NBII 2. comScore 3. FSMK 4. I3p 5. Optimizely
Skip add+tfidf	1. PRIVO 2. kathy 3. Elon 4. astronaut 5. cade
DEWE	1. Mavshack 2. Media Temple 3. Website 4. Epos Now 5. Compupress
DEWE+tfidf	1. Facebook 2. WhoSay 3. Blog 4. Instagram 5. Google+

Table 6: The performance of different models on link prediction and document classification.

Task	Link Prediction		Document Classification		
	MAP@ 10	Recall@ 10	Macro-F1	Micro-F1	Weighted-F1
Skip add	43.77%	20.03%	9.60%	21.22%	15.44%
Skip add+tfidf	47.36%	21.10%	14.09%	25.86%	19.98%
Skip mult	32.71%	14.71%	-	-	-
Skip mult+tfidf	10.88%	12.44%	-	-	-
PV	38.99%	19.22%	10.48%	23.21%	16.30%
DEWE	44.46%	20.84%	10.04%	23.30%	15.67%
DEWE+tfidf	<b>50.12%</b>	<b>23.51%</b>	<b>16.85%</b>	<b>29.97%</b>	<b>22.13%</b>

We achieve this by constructing the document embeddings of all documents first and then return the candidate documents which have the shortest cosine distance to a given document. All documents in the training set are used for evaluation, and the evaluation metrics we used are MAP and Recall. Since the embeddings trained by our model will contain the information from the referencing document, we hope that this information could help with the link prediction.

The task of document classification is to do a multi-class classification of the documents according to their categories in the Wikipedia. We assume that documents from the same categories are likely to reference each other. Since we map words to their definitions, these documents can be seen as containing the information of each other through the words with hyperlinks, so that they will be close to each other in the embedding space. To do the classification, we construct the document embeddings as above, and then feed them along with their labels to a supervised classifier. We use a 10-fold cross-validation over the entire training set for the evaluation via F1 Score.

The results on the two tasks are shown in Table 6. We ignore the results of **Skip Mult** on document classification for its poor performance. Our DEWE model achieves best performance on both tasks, it demonstrates that not only the word embedding got enriched by the definition, but also the document embedding. Based on all of the experiment above, we can conclude that our model is effective for both word and document modeling.

## 4 Related Work

Distributed word representations are first introduced by Rumelhart et al. (1988), and have been successfully used in a lot of NLP tasks, including language modeling (Bengio et al., 2006), parsing (Socher et al., 2013), disambiguation (Collobert et al., 2011), and many others.

Previously, word embeddings are often the by-product of a language model (Bengio et al., 2006; Collobert and Weston, 2008; Mikolov et al., 2010). However, such methods are often time consuming and involve lots of non-linear computations. Recently, Mikolov et al. (2013a) proposed two log-linear models, namely CBOW and Skip-gram, to learn word embedding directly from a large-scale text corpus efficiently. Glove, proposed by Pennington et al. (2014), is also an efficient word embedding learning framework, which combines the global word co-occurrence statistics as well as local context window information.

All of the methods mentioned above are mainly context-based models, and there exists some other works extending such models by using external knowledge. Bian et al. (2014) used semantic, morphological and syntactic knowledge to compute more effective word embeddings, Liu et al. (2015) used LDA to generate topical based word embeddings, and Rothe and Schütze (2015) used WordNet as a lexical resource to learn embeddings for synsets and lexemes. In this work, we improve the word embeddings by utilizing both context (extrinsic) and definition (intrinsic) information.

## 5 Conclusion

In this work, we learn word embeddings from both their intrinsic definitions and extrinsic contexts. Specifically, we extend the context-based model by mapping a word to its definition. Evaluations on four tasks demonstrates that our learning method is more effective than the previous context-based models.



## References

- [Bengio et al.2006] Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Frédéric Morin, and Jean-Luc Gauvain. 2006. Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186. Springer.
- [Bergstra et al.2010] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: A cpu and gpu math compiler in python. In *Proc. 9th Python in Science Conf*, pages 1–7.
- [Bian et al.2014] Jiang Bian, Bin Gao, and Tie-Yan Liu. 2014. Knowledge-powered deep learning for word embedding. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 132–148. Springer.
- [Blacoe and Lapata2012] William Blacoe and Mirella Lapata. 2012. A comparison of vector-based representations for semantic composition. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 546–556. Association for Computational Linguistics.
- [Bruni et al.2012] Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. 2012. Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 136–145. Association for Computational Linguistics.
- [Chen et al.2016] Jifan Chen, Qi Zhang, Pengfei Liu, and Xuanjing Huang. 2016. Discourse relations detection via a mixed generative-discriminative framework. In *Thirtieth AAAI Conference on Artificial Intelligence*.
- [Collobert and Weston2008] Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.
- [Collobert et al.2011] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.
- [Faruqui and Dyer2014] Manaal Faruqui and Chris Dyer. 2014. Community evaluation and exchange of word vectors at wordvectors.org. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, Baltimore, USA, June. Association for Computational Linguistics.
- [Faruqui et al.2016] Manaal Faruqui, Yulia Tsvetkov, Pushpendre Rastogi, and Chris Dyer. 2016. Problems with evaluation of word embeddings using word similarity tasks. *arXiv preprint arXiv:1605.02276*.
- [Finkelstein et al.2001] Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppín. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM.
- [Halawi et al.2012] Guy Halawi, Gideon Dror, Evgeniy Gabrilovich, and Yehuda Koren. 2012. Large-scale learning of word relatedness with constraints. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1406–1414. ACM.
- [Hill et al.2015] Felix Hill, Kyunghyun Cho, Anna Korhonen, and Yoshua Bengio. 2015. Learning to understand phrases by embedding the dictionary. *arXiv preprint arXiv:1504.00548*.
- [Hill et al.2016] Felix Hill, Roi Reichart, and Anna Korhonen. 2016. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*.
- [Hochreiter and Schmidhuber1997] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.
- [Hu et al.2014] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems*, pages 2042–2050.
- [Le and Mikolov2014] Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*, volume 14, pages 1188–1196.
- [Liu et al.2015] Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. 2015. Topical word embeddings. In *AAAI*, pages 2418–2424.

- [Mikolov et al.2010] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Interspeech*, volume 2, page 3.
- [Mikolov et al.2013a] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
- [Mikolov et al.2013b] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.
- [Mitchell and Lapata2010] Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive science*, 34(8):1388–1429.
- [Pennington et al.2014] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. *Proceedings of the Empirical Methods in Natural Language Processing (EMNLP 2014)*, 12.
- [Řehůřek and Sojka2010] Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May. ELRA. <http://is.muni.cz/publication/884893/en>.
- [Riedel et al.2013] Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas.
- [Rothe and Schütze2015] Sascha Rothe and Hinrich Schütze. 2015. Autoextend: Extending word embeddings to embeddings for synsets and lexemes. *arXiv preprint arXiv:1507.01127*.
- [Rumelhart et al.1988] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1988. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1.
- [Shaw et al.2013] Ryan Shaw, Anindya Datta, Debra VanderMeer, and Kaushik Dutta. 2013. Building a scalable database-driven reverse dictionary. *IEEE Transactions on Knowledge and Data Engineering*, 25(3):528–540.
- [Socher et al.2011] Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, pages 801–809.
- [Socher et al.2013] Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. 2013. Parsing with compositional vector grammars. In *ACL (1)*, pages 455–465.
- [Yang and Powers2006] Dongqiang Yang and David MW Powers. 2006. *Verb similarity on the taxonomy of Word-Net*. Masaryk University.
- [Zock and Bilac2004] Michael Zock and Slaven Bilac. 2004. Word lookup on the basis of associations: from an idea to a roadmap. In *Proceedings of the Workshop on Enhancing and Using Electronic Dictionaries*, pages 29–35. Association for Computational Linguistics.